USC University of Southern California

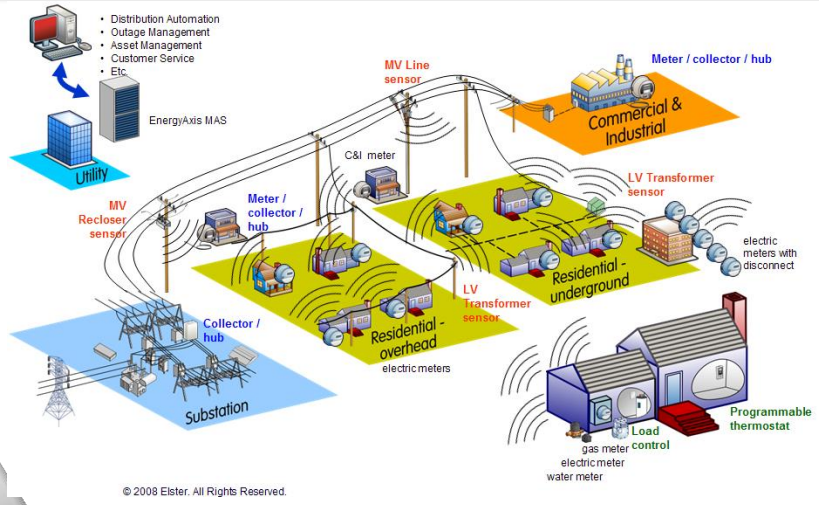
USC Viterbi School of Engineering

GoFFish : A Sub-Graph Centric Framework for Large-Scale Graph Analytics

Charith Wickramaarachchi,

Yogesh Simmhan,Alok Kumbhare, Soonil Nagarkar,Santosh Ravi,
Marc Frincu, Cauligi Raghavendra, Viktor Prasanna

Big Data



Twitter Storm

Pregel

Map Reduce

1st Mentioning of Big Data

Michael Cox and David Ellsworth publish "Application-controlled demand paging for out-of-core visualization" in the Proceedings of the IEEE 8th conference on Visualization, 1997

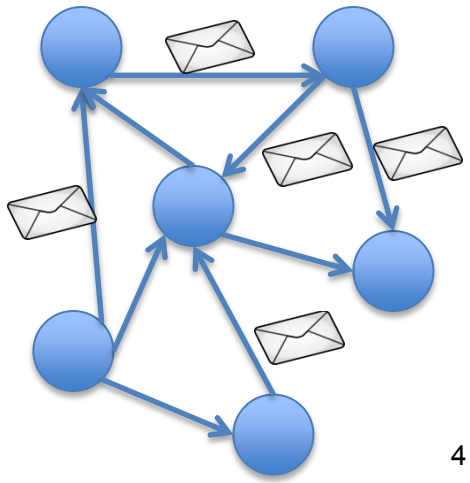
http://www.gartner.com/it-glossary/big-data/

GoFFish Overview

- Scalable platform for ***large scale graph analytics*** in commodity clusters and cloud.
- Simple yet scalable programming abstraction for large scale graph processing
- More than **10x improvements** in some graph algorithms over traditional graph processing systems
- Components
  - **GoFS** : Sub-graph centric ***Distributed Graph Storage***
  - **Gopher** : Distributed ***BSP-based programming abstraction*** for Large Scale graph analytics framework

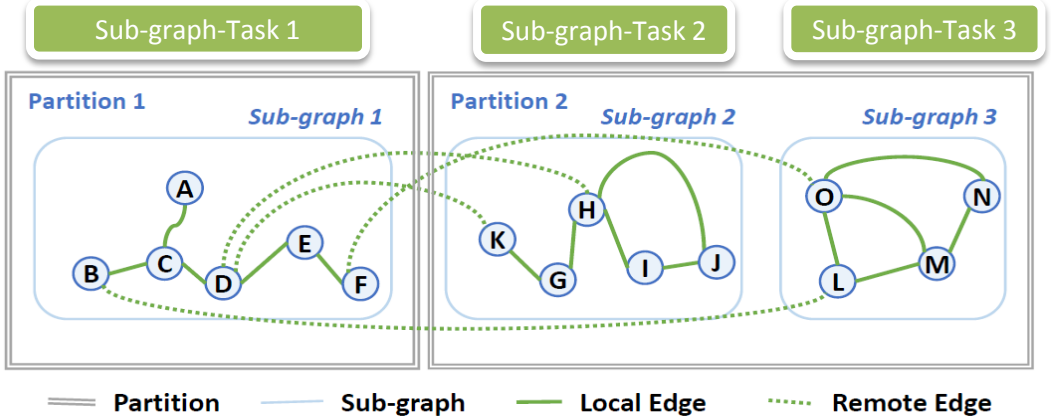
Vertex centric programming model

- Iterative vertex centric programming model based on Bulk synchronous parallel model.
- In each iteration vertex can
  1. Receive messages sent to it in previous iteration
  2. Send messages to other vertices
  3. Modify its own state
- Vertex centric programming mode- ***Think like a vertex***
- Very simple,
- High communication overhead
- Pregel, http://giraph.apache.org/



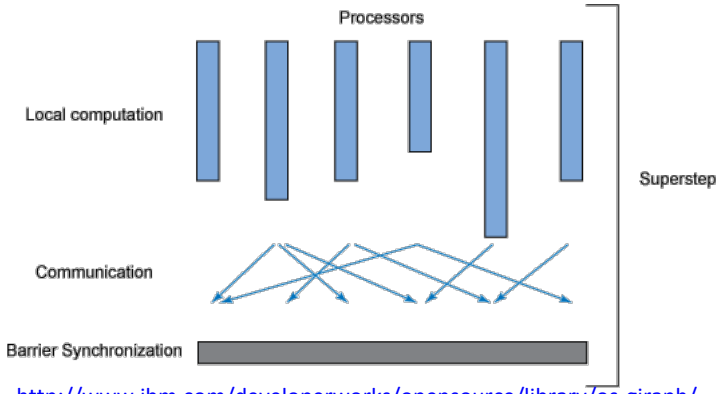
Subgraph Centric Programming Model

- Partition graph in to set of sets of vertices and edges
- ***Sub-graphs*** – Connected components in a partition.
- User logic operates on a sub graph
  - Independent unit of computation
- Resource allocation
  - Single Partition → Single Machine | Single Sub-graph → Single CPU
- Data loading
  - Entire partition is loaded into memory before computation
  - Tasks retain sub-graphs in memory within the task scope

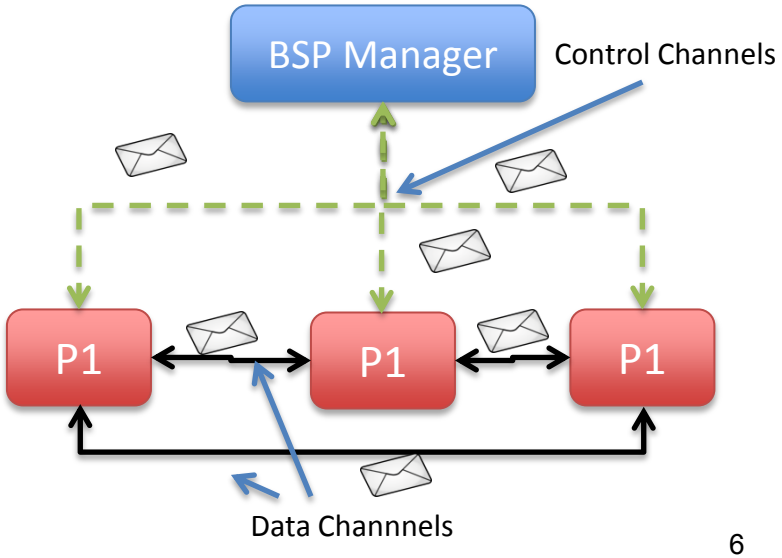


Sub-graph Centric Programming Model

- **Algorithm** : Seq^n of super-steps separated by global barrier synchronization
- Super Step i:
  1. Sub-graphs compute in parallel
  2. Receive messages sent to it in super step i-1
  3. Execute same user defined function
  4. Send messages to other sub graphs (to be received in super step i+1)
  5. Can vote to halt : I'm done / de-activate
- Global ***Vote to Halt*** check
  - termination → all sub-graphs voted to halt
- Active sub graphs participate in every computation
- De-activated sub-graphs will not get executed/activated unless it get new messages



http://www.ibm.com/developerworks/opensource/library/os-giraph/

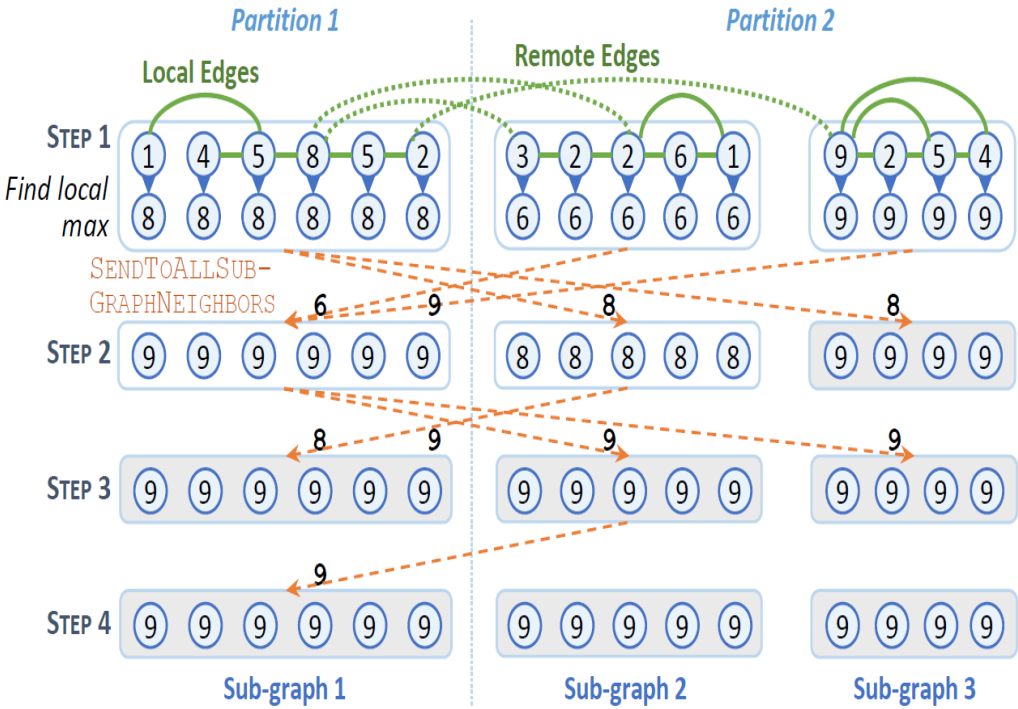


Sub-graph Centric Programming Model Example

- **Algorithm** – Max vertex value
- **Input** – Connected graph with different vertex values $\{n_1, n_2 \ldots n_i\}$
- **Output** – Each vertex with the Max ($\{n_1, n_2 \ldots n_i\}$)

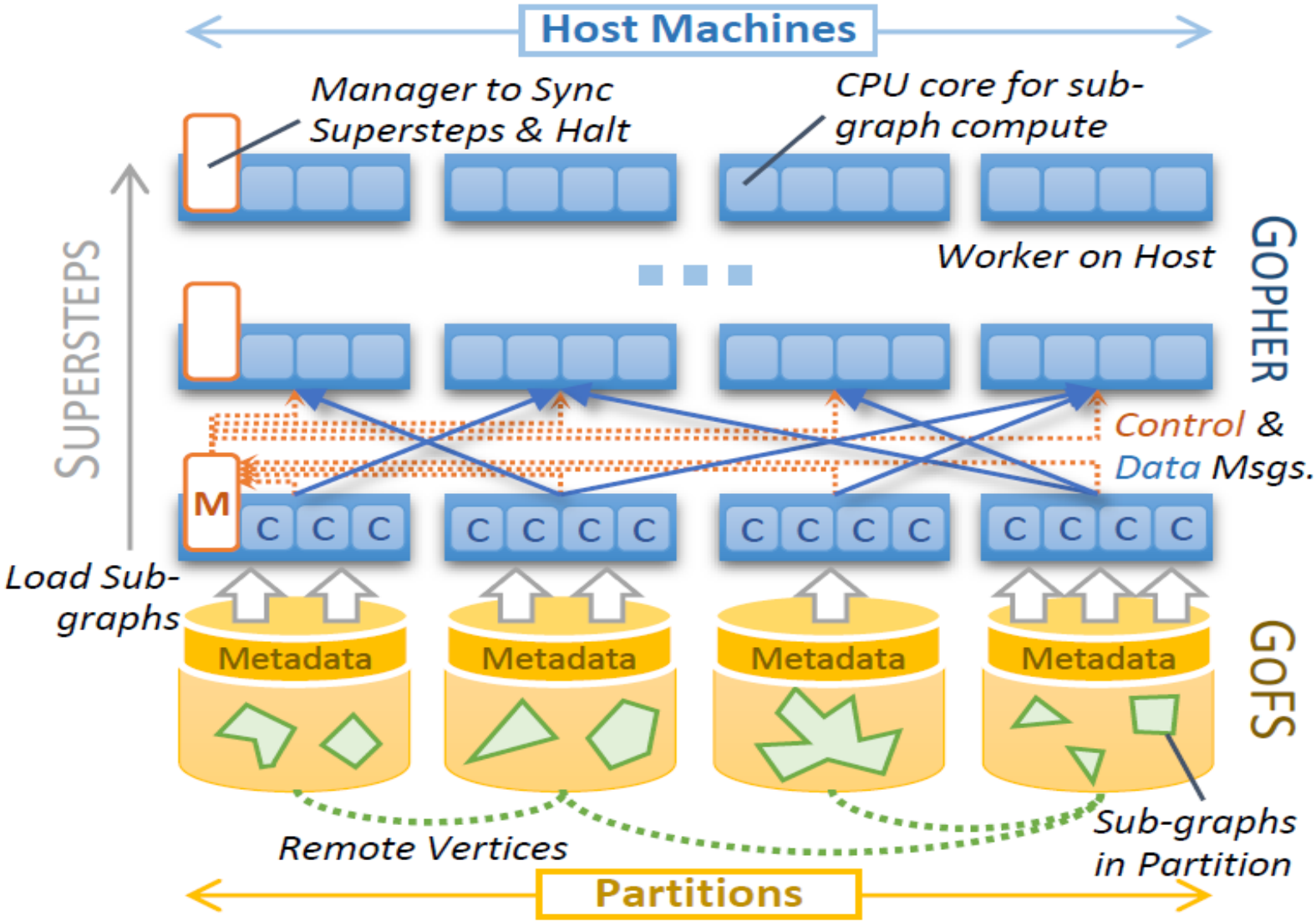
Algorithm 1 Max Vertex using Vertex Centric Model

```
1:  procedure COMPUTE(Vertex myVertex, Iterator⟨Message⟩ M)
2:      hasChanged = (superstep == 1) ? true : false
3:      while M.hasNext do            ▶ Update to max message value
4:          Message m ← M.next
5:          if m.value > myVertex.value then
6:              myVertex.value ← m.value
7:              hasChanged = true
8:      if hasChanged then            ▶ Send message to neighbors
9:          SENDTOALLNEIGHBORS(myVertex.value)
10:     else
11:         VOTETOHALT( )
```



- Compared to Vertex centric model
  - Less communication
  - Fast convergence

Goffish Architecture



Performance Comparisons – Experimental Setup

- Cluster
  - 12 Nodes, 8-Core Intel Xeon CPU (each)
  - 16GB RAM (each), 1TB HDD (each)
- Network
  - Gigabit Ethernet
- Apache Giraph latest version from trunk
  - Includes Performance improvements from Facebook !

Data Set	Vertices	Edges	Diameter
RN: California Road Network	1,965,206	2,766,607	849
TR: Internet path graph from Tracesroutes	19,442,778	22,782,842	25
LJ: LiveJournal Social Network	4,847,571	68,475,391	10

Runtime Performance Comparison with Vertex Centric Model

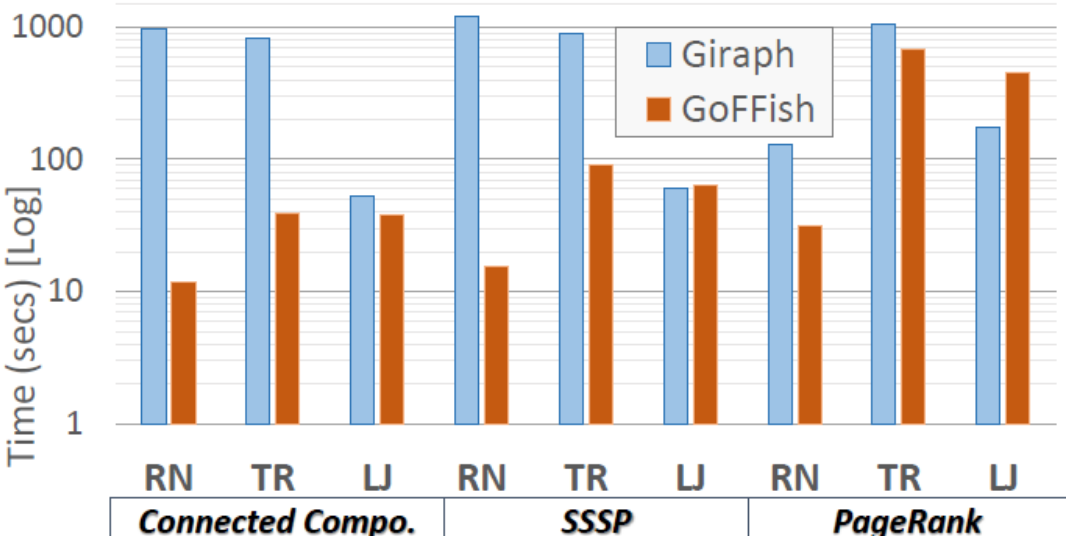
- Connected components
  - 81x improvement using California Road Network (RN) dataset
  - 21x improvement using a Trace route path network of a CDN. (TR) dataset
- Single Source Shortest Path
  - 32x improvement using RN dataset
  - 10x improvement using TR dataset

Page Rank

4x improvement using RN dataset

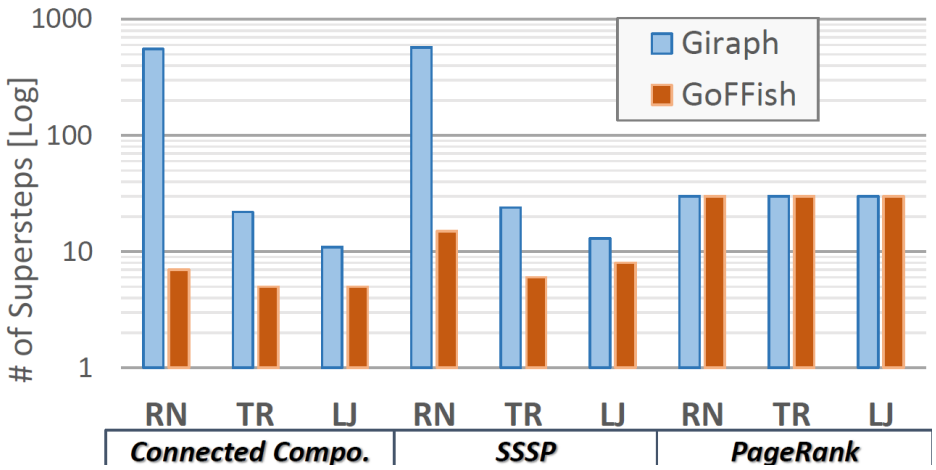
1.5x improvement using TR dataset

Not an ideal algorithm for sub-graph centric programming model



Super Step Reduction Comparison with Vertex Centric Model

- Connected components
  - ~79x reduction using RN dataset.
  - ~4x reduction using TR dataset
  - ~2x reduction using Live Journal dataset (LJ)
- Single source shortest path
  - ~38x reduction using RN data set
  - 4x reduction using TR dataset
  - ~1.6x reduction using Live journal dataset



Conclusion

- Introduced a ***sub-graph centric programming abstraction*** for large scale graph analytics on distributed systems
  - Simple
  - Enable using shared memory algorithms at sub-graph level.
- ***Sub-graph centric algorithms*** and performance results
  - Connected Components
  - Single Source Shortest Path
  - Page Rank
- Issues and Future work
  - Sub-graph aware partitioning
  - Sub-graph centric algorithms
- Try now
  - https://github.com/usc-cloud/goffish

Thank you!

http://thesciencepresenter.wordpress.com/category/behaviour-management/